Flight Software Branch
Flight Software Test Bed
Requirements Guidelines

May 21, 2001

DRAFT

1 Introduction

Flight software is inherently difficult to validate. Integration and Test activities and post-launch flight operations rely to a much greater extent on flight software performance than hardware engineers and managers sometimes appreciate. Flight software teams are severely handicapped and their work is greatly complicated when Project management avoids early investments into adequate FSW test-bed environments. When investment is limited, lack of FSW readiness for I&T and operations will often be the cause of politically embarrassing and costly schedule delays – typically with a corresponding late investment into FSW test-bed fidelity.

This document establishes general requirements for flight software testbed systems. Justifications for the requirements are based on years of Lessons Learned and repeated experiences of using both quality and sub-quality test environments. Careful consideration of the data in this document and investment into the highest fidelity test-beds feasible, will mitigate significant risks to flight software and flight hardware schedules, plus the operations and science objectives of the mission.

Why is it so difficult to test FSW?

- complexities of the post-launch spacecraft environment and operations scenarios
- the amount of special purpose flight hardware to be managed by FSW
- the criticality of time synchronization among the full range of onboard activities the unique characteristics of the actual flight data system hardware in which the FSW resides and communicates

In addition to supporting flight software test and validation roles, flight software test-environments are necessary for several different activities throughout the life of the mission, including:

- Flight Hardware Interface Testing
- Flight Data System functional and performance testing
- Flight Operations Team Training
- Flight Operations Procedure Development and Validation
- Ground Systems Development and Test

The focus of this document is FSW testbeds for use by the FSW teams (FSW development, test, maintenance). – talk about the need to identify special reqmts. for any of these other purposes early in mission design and get those reqmts. accommodated in the plan.
Eg. of special reqmts. for those teams

I couldn't fit this anywhere useful , seems too vague for the range of testbeds we decided to discuss and incomplete.

These are often competing activities and sufficient physical facilities (test beds) must be provided to support them all during parallel schedules. Make recommendations with justifications.

It is intended as guidance for personnel involved in the specification, design, evaluation(?), and development of flight software test beds.

2 Flight Software Test Bed Overview

A generic high fidelity Flight Software Test Bed (FSTB) system contains the following subsystems (see figure 1).

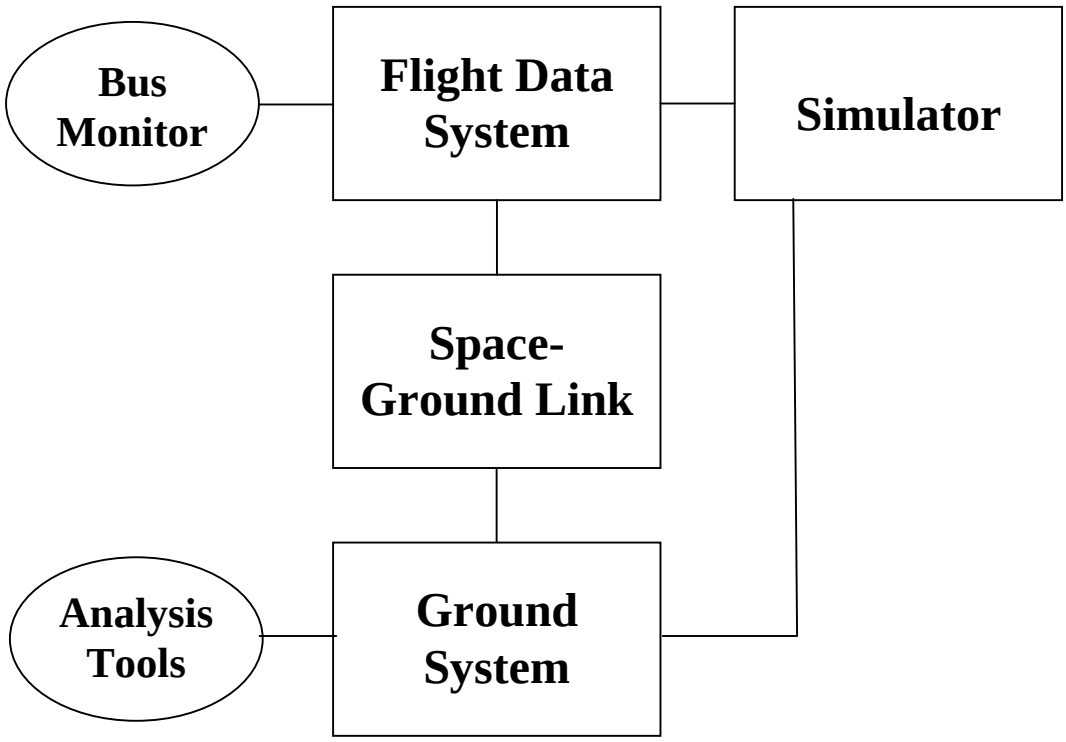

Figure 1 - Flight Software Test Bed Subsystems

Flight Data System	The on-board processor(s), the flight software, and the data buses and interfaces to the other spacecraft systems. These may be actual flight components, or Engineering Test Units. The Flight Data System also includes the interface to the Simulator. This interface is a combination of the spacecraft data bus, and direct analog or digital connections.
Simulator	Provides a software simulation of the spacecraft operating environment including (but not limited to) position and velocity, attitude sensor input data, solar and lunar position, star positions, earth magnetic field, and thermal models, etc. Also models any flight components not present in the Flight Data System, such as sensors, actuators and instruments.
Space-Ground Link	Provides the link between the Flight Data System and the Ground System. This is a simulation of the real space to ground command and telemetry link; it may simulate the full radio frequency link features, or be as simple as a direct Ethernet connection.

Ground System	Provides a command and telemetry interface to the Flight Data System. This component also hosts other control center functions such as a test and operations language processor and a data archive and associated data analysis tools. Ideally, this is the same ground system used in the actual flight control center. In fact, for full fidelity mission simulations, the actual flight control center can be used for the Ground System of the test bed. The Ground System also controls the Simulator, and all other aspects of the test bed. Thus there is a single point of configuration control for all tests, and all tests can be run from the single Ground System console.
Bus monitor	Enables the tester to examine traffic on the Flight Data System buses, and insert data either for simulation or bus error testing.
Analysis Tools	Analyses data from tests, to verify flight system performance.

3 General Flight Software Test Bed Requirements

[3.0.1] The operator shall be able to configure and run a complete test or simulation from the ground system console. This requires that the ground system, the simulator, and the space-ground link be able to be configured and run from the ground system console.

Justification:

1. Enables consistent coordination between different elements of the test bed.
2. Allows test runs to be controlled from a single automated script, thus increasing repeatability and reliability.

[3.0.1.1] The test bed shall be able to set and synchronize spacecraft clock time, model simulation time, and ground system time. Synchronization tolerances will vary by mission, depending on the required accuracy of the navigation models. Generally, the Flight Data System will provide time to the other subsystems. Both absolute and relative synchronization shall be supported.

[3.0.1.1.1] Absolute synchronization shall set all subsystem clocks to the same absolute time.

[3.0.1.1.2] Relative synchronization shall keep all subsystem clocks counting at the same rate, but allow different absolute times.

Justification:

1. The Dynamic Simulator must know the spacecraft clock time to provide correct positions for planets, stars, etc.
2. The Ground System must know the spacecraft clock time to provide telemetry time tags.
3. Providing the spacecraft clock time in the dynamic simulator telemetry allows it to be synchronized with spacecraft telemetry.
4. Relative synchronization provides support for tests which require a warm start of the flight computer; during this time the flight software is providing an incorrect absolute time.

[3.0.1.2] Telemetry from the dynamic simulator, also referred to as truth data, shall be collected in real time by the Ground System, in a manner similar to the way normal Flight Data System telemetry is collected.

Justification:

1. Allows for a central point of collection for all data, using a single format.
2. Allows for real-time comparison of spacecraft telemetry and simulator telemetry.
3. Automated scripts can wait and take action based upon both spacecraft telemetry and simulator telemetry values.
4. Allows the logging function to be off-loaded from the dynamic simulator.

[3.0.2] The following documentation shall be provided with the test bed:

1. Spacecraft and Dynamics model requirements and algorithms
2. System Test and Operations Language (STOL) programming guide
3. Operations and Maintenance guides for each FSTB component
4. Development Tools manuals
5. Analysis Tools manuals
6. Display Language Guide for control center simulator
7. Data Base Manual for control center simulator
8. Global Test Set requirements document

Justification: Self explanatory – lack of proper documentation will result in steeper learning curves and loss of efficiency.

[3.0.3] FSTB System configuration shall be controlled via configuration files that can be saved and edited.

Justification:

1. Allows new runs to be quickly and consistently initiated.
2. Makes variations of the same basic test easy to set up.
3. Test repeatability is improved
4. If the history of these files is properly maintained, regression testing is supported more easily.

4 Flight Data System Requirements

There are various types of flight data system hardware (flight processors, data busses, i/o hardware, uplink/downlink cards, etc.) that could be a part of a flight software test bed. Each has particular purposes, and there are pros and cons associated with each. These include:

1. Engineering Models (EMs) – Actual flight qualified hardware. Same exact hardware that is used on the spacecraft.
2. Engineering Test Units (ETUs) – The same hardware as that used on the spacecraft, meaning the same parts and board layout and the having the same form, fit, and functionality, except that it is not flight qualified.
3. Bread Board (or Brass Board) – Same chip set and same functionality as the flight hardware, but not the same board layout. Worth noting is that for the 1750A flight processor, Honeywell makes what they call a Software Development Unit (SDU). This is a brass board flight processor, except it contains extra facilities to support software development. These facilities allow for symbolic debugging and trap and trace capabilities. SDUs are a must for FSW maintenance. May enable the debugging of problems that may not be solvable without these capabilities, or at least greatly reduces the time it takes to debug problems and thus reduces cost and risk. A little more expensive, but well worth the cost. If the flight computer vendor provides these capabilities, then at least one test bed should contain them, and the software developers and maintainers need to have access to it.

Also discuss how much actual spacecraft flight hardware ought to be a part of the test bed and how much could/ought to be simulated. For instance the test system used by the FOT may greatly benefit from having actual Solid State Recorders (SSRs), Uplink/Downlink equipment. Also what are the pros and cons of having an actual ACE?

Flight Software Test Beds can come in a number of configurations depending upon how much of the spacecraft flight hardware is actually present and how much is simulated. This could range from the actual spacecraft connected to a stimulator (as in I&T), down to a purely software spacecraft simulator. These different test bed configurations are used for different purposes and are developed at various stages in the spacecraft's development cycle. These configurations are summarized in the following table ranging from the most flight hardware in the system to the least. Note that while the requirements described in this document are most relevant to configurations three through five, a number of the requirements still apply to the other configurations.

Configuration Description	Purpose
1. Full spacecraft with stimulator	Spacecraft I&T
2. Full spacecraft with i/f to dynamics models simulator	Spacecraft I&T
3. Complete spacecraft flight data system (containing all flight CPUs and FSW), flight sensors and actuators, other flight hardware (including Attitude Control Electronics, if	

Configuration Description	Purpose
the spacecraft has one), hardware i/f, and dynamics models simulator	
4. All flight CPUs, FSW, flight hardware (including ACE, if the spacecraft has one), hardware i/f, and simulator (dynamics models and flight hardware models, including sensor and actuator models)	System level test, On-orbit FSW Maintenance
5. Individual flight CPU with its flight software, simulator, simulator interface (all other CPUs are simulated)	FSW integration and test, Build-level test
6. FSW and simulator running together on the same desktop computer (can run faster than real time)	Initial FSW development, Unit test
7. High fidelity simulator with no FSW (can run faster than real time)	Analysis

[4.0.1] The Flight Data Systems processor(s) shall be the same make and model as that used on the spacecraft, although it need not be flight qualified. All data buses, memory types and sizes, I/O ports, clocks, and other supporting devices must also be electrically equivalent to the flight data systems hardware.

Justification:
1. Variation from the exact characteristics of the flight processor and supporting devices increases the risk that Test Bed functionality and performance do not match on-orbit Flight Data Systems performance.
2. Problems that occur in the spacecraft flight data system should be able to reproduced in the test bed flight data system for trouble shooting purposes.

[4.0.2] The FSW executables should be exactly the same as used on the spacecraft. The FSW should not need to be modified to interact with the other elements of the FSTB.

Justification: Variation from the exact configuration of the flight software increases the risk that on-orbit performance does not conform to Test Set performance.

[4.0.3] The test bed flight data system shall contain any redundant components that are present on the spacecraft.

Justification: Allows for the testing of processor swap/failover management.

5 Simulator Requirements

The spacecraft simulator simulates the spacecraft beyond the flight data system. It consists of two primary elements: the dynamics models and the flight hardware models. The dynamics models provide a simulation of the spacecraft's operating environment in terms of:

- Position and velocity;
- Attitude;
- Vehicle rotation rates and accelerations;
- Temperature;
- Magnetic field strength and direction;
- Atmospheric drag;
- Gravity gradients;
- Solar radiation flux effects;
- Sun spot modelling;
- Geomagnetic activity index;
- Spacecraft component cryogenic venting;
- Etc.

This data is also provided in the form of inputs to the Flight Hardware models. The dynamics models also receive inputs from the actuator models in the Flight Hardware models. The fidelity of the dynamics models should be as high as practical, but consistent with the precision and fidelity of the FSW requirements.

The Flight Hardware models provide a simulation of spacecraft subsystems, including sensors, actuators, spacecraft power system, thermal control system, payload or instruments, and communications system. Components that have been simulated include the following:

- Attitude and guidance sensors:
  - Inertial Reference Units (IRU) and gyros;
  - Coarse Sun Sensors (CSS);
  - Fine Sun Sensors (FSS) or Digital Sun Sensors (DSS);
  - Earth Sensors (ES);
  - Star Trackers (ST);
  - Three-Axis Magnetometers (TAM) or Magnetic Sensing Systems (MSS);
  - Global Positioning System (GPS);
- Attitude control actuators:
  - Reaction Wheel Assemblies (RWAs) (including reaction wheel tachometers);
  - Thrusters;
  - Magnetic Torque Bars (MTB);
  - Propulsion Tank (pressure, mass moments);
- Spacecraft power system:
  - Solar Array power elements;
  - Batteries;
  - Battery power regulating elements;

  - Solar Array Drives;
- Thermal control system:
  - Thermal control sensors (temperature sensors);
  - Thermal control active components (heaters, pumps);
- Steerable antennae;
- Instrument mechanisms;
- Deployable components; and
- Interface electronic models (e.g. Attitude Control Electronics)

Each component must be simulated with at fidelity that allows the Flight Data System to be fully tested, in both nominal operation and failure detection and handling. For example, if a component normally provides telemetry that confirms a command, that telemetry must be simulated. Payload or instrument simulations are usually provided to a lower level of fidelity to include unmodeled instrument telemetry and only command counter and telemetry verifier response to commands. Instruments that have characteristics that significantly affect spacecraft dynamics or operation (large moving parts, large power draw, high data rates in some modes, etc.) should be more fully modeled.

In most cases the entire simulator resides on a single computer. However, often is useful to have certain spacecraft models hosted on their own separate computer or rack. An example of this might be a separate power or thermal simulator. The advantage of this is that these systems can be developed separately and delivered to their respective flight subsystems for independent tests. In cases such as this the main part of the simulator must be able to communicated with subsystem simulator so that the entire simulation can be coordinated.

[5.0.1] The simulator shall execute in “hard real time,” that is it must be deterministic and never miss an event.

Justification: Data bus schedules and many FSW control algorithms, such PID, require deterministic behavior. If an event were missed, the data collected would be incorrect.

[5.0.2] The simulator host computer must be powerful enough to execute the model cycle in real time. That is its CPU must be fast enough and its operating system capable enough to provide all required data at a rate and with the timing accuracy that the flight data system requires.

Justification: Self explanatory – models must execute on the same time scale as the FSW. Note this will require that analysis be done early in the design of the Flight Software Test Bed to determine what the processor capabilities will need to be. Capabilities that must be taken into consideration are CPU utilization and event time accuracy (both input/output and latency). A computer should be chosen that will exceed the capabilities determined by the analysis with at least a 50% margin.

[5.0.3] The units of measurement used in the simulator shall match those used in the FSW. Preferably this will be *Systeme Internationale*.

Justification:
1. Must have consistent units and the FSW is the driver.
2. SI units are easier to understand and more standard.

[5.0.4] The simulator shall drive all data via the flight hardware models that the flight data system samples, and receive all signals output by the flight data system, except those involved in the space-ground link.

Justification: This allows for testing of all of the FSW. All data that gets input to the flight data system has the potential to be acted upon by it, and so must be provided.

[5.0.5] The simulator shall model all effects that the FSW can react to.

Justification: This allows for testing of all of the FSW.

[5.0.6] It shall be possible to disable all dynamics and flight hardware models. The output of disabled models shall be set explicitly by the user, in either engineering units or hardware counts.

This provides an "open loop" testing capability, i.e., the Flight Data System can be operated without input or feedback from simulator models. This mode of operation effectively makes the simulator into a stimulator/monitor.

Justification:
1. Some testing requires directly setting simulator outputs that must not be overwritten by models.
2. Diagnosing some problems is simpler if simulator models can be turned off so the outputs are stable.
3. This allows for direct reading of spacecraft hardware outputs.

[5.0.7] The dynamics models should allow the user to start the simulation at any point in the currently defined orbit and at any attitude defined by the user.

Justification: Many tests require the modeled spacecraft to be in a specified position and attitude. Waiting for the models to propagate to such conditions may be impossible or unreasonably time consuming.

[5.0.8] The simulator shall be able to accept commands from the ground system during the simulation run.

Justification: This allows an entire test run to be controlled from the ground system, and thus allowing for coordination of the elements of the test (FSW, simulator, and ground system test script). If the simulator could not accept ground system commands, then

running a test run would require activities be performed on both the ground system and the simulator console. This would diminish the repeatability of tests because there would be no guarantee that the separate activities could be timed relative to one another in the same way during different test runs.

[5.0.9] It shall be possible to re-initialize the conditions for all models, to any value valid for the model, without having to restart the simulator.

Justification: This allows the user to perform simulation runs back to back. Multiple test procedures can be grouped together and run as a single batch without operator intervention. Usually, a test procedure is utilized to restore the simulator to a known state.

[5.0.10] The simulator shall allow the user to set simulator parameters during runtime without having to stop the simulation. The method of setting simulation parameters shall be the scale-factor and bias method, i.e. desired_value = scale_factor * simulator_parameter + bias. This capability shall also be provided as a "set_parameter" process to set simulation parameters prior to the simulation control cycle, and as a "set_telemetry" process to set simulation parameters after the simulation control cycle.

Justification:

1. Gives the user the ability to control the values from the simulator into the FSW for the purpose of testing and trouble shooting.
2. Allows for a simple way of implementing failure conditions in the flight hardware models. Thus a number of the failure conditions do not have to be hard coded into the simulator. Since it is not possible for the simulator designer to anticipate all failure conditions ahead of time, this feature allows the user to control simulator outputs in order to create any desired failure condition (or nominal condition for that matter) in order to fully test the flight software.
3. Having the ability to control inputs into the simulator allows the user to manipulate the models in a way that can be useful for flight software testing.

[5.0.11] The simulator shall have a scripting ability that at a minimum allows the execution of scripts that contain sequences of commands with relative and absolute time waits. Additional scripting could be provided to allow the implementation of simple models without having to recompile the simulator.[1]

Justification:

1. Allows for more accurate timing of simulator command sequences than if sent by the ground system. (Note: Requirement [5.0.8] implies that simulator script execution may be initiated from the ground system.)
2. Allows for significantly faster processing of large simulator command sequences (particularly large initialization scripts) than if sent by the ground system.

[1] Do we want to add that more than one script can run concurrently? This would allow simple models to run in the background.

3. The ability to implement simple model allows the user to create models that were not designed into the system. For example there may not have been a need for the simulator to include a sophisticated temperature control system model. With the ability to implement simple models, a tester can set up a script that, for instance, causes thermistor readings to ramp up when the FSW commands associated heaters to turn on. This is particularly important if the user is not provided with the source code or does not have the coding expertise to make the desired changes.

[5.0.12] A separate logging capability needs to be provided in the simulator in order to facilitate the capturing of high-frequency data.

Justification: Because there may be limitations on the frequency at which the ground system can receive data from the simulator, a separate logging capability is necessary for capturing high frequency simulator data. (See Requirement [3.0.1.2] .)

[5.0.13] The simulator host machine should be able to record and archive all simulator truth data for long-duration simulations (at least 48 hours).

Justification: This supports the requirement for long duration tests.

[5.0.14] The simulator shall archive truth data in ASCII format. Most test analysts will transfer the archived data to desk top computers using a variety of mathematics and plotting packages for analysis.

Justification: Since ASCII format is portable, this allows the archived data to be transferred to other computer systems where it can be analyzed using a variety of mathematics and plotting packages.

5.1 Dynamics Models Requirements

[5.1.1] The orbit model, with disturbances enabled, should be sufficiently accurate and stable so that mission position and velocity tolerances can be maintained over the period between mission orbit updates.

Justification: FSW performance and stability tests over a long period of time are a necessary requirement. To test FSW responses to magnetic field variations, for example, a test that traverses enough of the magnetic field to provide an adequate sampling of the B field vectors will take several orbits, depending on the mission orbit design. Some modeled variables, like gravity gradient, have secular terms that build up over a number of hours, requiring long duration tests.

[5.1.2] The attitude sensor data models should be sufficiently accurate and stable so that mission pointing tolerances can be maintained for long-duration simulations (at least 48 hours).

Justification: FSW performance and stability tests over a long period of time are a necessary requirement.

[5.1.3] It shall be possible to disable complex effects[2] that are modeled by the dynamics models (such as atmospheric drag or orbit model disturbances).

Justification:

1. In some situations, analysis of test data will only be possible or easier if the random effects of disturbances and noise can be ignored.
2. Some flight software modules are designed to deal with only one type of disturbance, e.g., gravity gradient. Disabling the other disturbances allows the tester or trouble-shooter to focus on just the module in question.

[5.1.4] The dynamics models shall include the following mission-specific features:

[5.1.4.1] For low earth orbit missions, the orbit model should account for aerodynamic drag, the dominant disturbance in Low Earth Orbit.

Justification: Variations in the modeled orbit due to drag are necessary to fully test navigation software.

[5.1.4.2] The orbit model shall account for disturbances due to solar radiation pressure.

[5.1.4.3] The orbit model should allow the user to propagate orbit either by time propagation from orbital elements or propagation of spacecraft states using a gravity model. The gravity model in the simulation should be more precise than that used by the FSW so that position and velocity errors can be more closely approximated.

Justification: The Navigation software on many spacecraft uses state vector propagation or interpolation as the prime mode with orbit element generation as a backup. State vector propagation methods are particularly useful for missions orbiting the Lagrange points and accounting for multi-body gravitational forces. The simulator needs to provide truth data of the same precision as the flight software methods.

[5.1.4.4] The dynamics model shall model the attitude effects of thruster firings.

[5.1.4.5] The dynamics model shall model the orbit effects of thruster firings.

[5.1.4.6] The orbit model shall provide vectors from the spacecraft to the TDRS satellites and the NASA earth communications stations.

Justification:

1. This capability is required for testing antenna pointing software.

[2] Shouldn't we list what these complex effects are, just like we list noise and systematic effects in the next section.

2. It allows the simulation system to model the communications duty cycles for the mission.
3. It allows the verification of FSW-generated TDRS ephemeris.

[5.1.4.7] The dynamics models shall model effects of gravity gradients, especially for gravity stabilized missions.

Justification: This has a significant effect on attitude.

[5.1.4.8] The earth magnetic field model shall conform to the International Geomagnetic Reference Field coefficients model maintained by Flight Dynamics at GSFC.

Justification: Standard representation of the earth magnetic field model provides the most realistic testing of the TAM and magnetic torque bar control software.

[5.1.4.9] The magnetic field model should be at least as granular as the flight software model.

Justification: Reasonable estimation of errors requires that the simulated truth data be more granular than the FSW data being evaluated. Good representation of the magnetic field is particularly important for missions using magnetometers and magnetic torquer bars for either primary or back up attitude control.

[5.1.4.10] The simulator shall provide solar, lunar, and planetary ephemeris models of sufficient accuracy to meet the mission navigation system requirements.

Justification:
1. Realistic navigation and ACS software testing requires this capability. Precision requirements are necessary for error estimations.
2. Needed to test FSW occultation models.

5.2 Flight Hardware Models Requirements

[5.2.1] The Flight hardware models shall model all flight components, including redundant components, that are relevant to the FSW and that are not actually present in the test bed flight data system.

Justification:
1. Necessary for full testing of the FSW and to support full mission scenario simulation.
2. Redundancy simulation is required for proper testing of failovers and contingency responses.

[5.2.2] The Flight Hardware models shall simulate the nominal operation for the modeled components, with the fidelity required by the Flight Data System nominal operations.

Justification: In some situations, an adequate "model" of a particular feature is to just provide a user-settable telemetry point, rather than to try to implement a fully realistic model of the component. The goal is to test the Flight Data System.

[5.2.3] The flight hardware models shall correctly model the timing of component responses, with the fidelity required by test the Flight Data System. This includes power up cycles, initialization delays, spin up delays, etc.

Justification:

1. Required to verify that FSW timing problems are not present.
2. This makes simulation more realistic.

[5.2.4] The flight hardware model outputs shall use the same bit quantization as the actual flight hardware.

Justification: If the bit quantization of the flight hardware model had a different granularity than the actual flight hardware, the data supplied to the test bed flight software may cause behavior different than that on the actual spacecraft.

[5.2.5] Any systematic effect that is modeled in Flight Software (e.g., sensor and actuator misalignments, gyro drift biases, MTA- TAM coupling, etc.) shall be modeled in the flight hardware models. The simulator models should have a fidelity level equal to or better than the flight software models.

Justification:

1. If the FSW compensates for an effect that the simulator does not model, then there is no way of determining that it is performing correctly.
2. The simulator truth data must be at least as good as the flight model being tested or no valid conclusions can be made regarding the correctness of the flight model.

[5.2.6] The flight hardware models shall model all noise sources and systematic effects that can affect system performance.

Justification: The Flight Data System must be able to accommodate noisy sensor data and systematic effects associated with hardware components. Noisy data is necessary to test Kalman Filters.

The flight hardware models shall model the following mission-specific noise sources and systematic effects[3]:

[5.2.6.1] The IRU and Gyro models shall model:

- Random Walk;
- Random Noise;
- Drift;

[3] What about justifications for all of these?

- Temperature induced effects; and
- Saturation (Spacecraft moving faster than gyro can sense).

[5.2.6.2] The Coarse Sun Sensor model shall model:

- Noise on current;
- Albedo from planetary objects;
- Occlusion from movable h/w;
- Field of View; and
- Non-linearities.

[5.2.6.3] The Fine Sun Sensor/Digital Sun Sensor model shall model:

- Noise on current;
- Albedo from planetary objects;
- Reflection off other s/c parts
- Quantization;
- Non-linearity in calibration; and
- Detection Threshold.

[5.2.6.4] The Earth Sensor model shall model:

- Noise on current;
- Earth oblateness[4]; and
- Horizon radiance.

[5.2.6.5] The Star Tracker model shall model:

- Gaussian noise;
- Errors in magnitude (bias);
- Velocity Aberration;
- Error is Star Tracker software;
- Lunar and planet occultations; and
- Spoiler objects.

[5.2.6.6] The Three-Axis Magnetometer/Magnetic Sensing System model shall model:

- Noise on output components;
- Contamination from Magnetic Torque Bars;
- Residual spacecraft magnetic dipole[5]; and
- Quantization.

[5.2.6.7] The reaction wheel model shall model:

- Bearing stiction and linear friction;
- Tachometer noise; and

[4] Should this be a part of the dynamcis simulator? Yes. What about the horizon radiance?

[5] This parameter is sometimes determined by ground system software, so the ability to model this effect is also useful for ground systems testing.

- Reaction wheel motor drive back electromagnetic force.

[5.2.6.8] The thruster model shall model centroid misalignment and cold/hot running.

[5.2.6.9] The magnetic torque bar model shall model:

- Noise on measurement;
- "Overshoot" effect if current changes in a step-wise fashion;
- Effect on magnetic field of MTB operations; and
- MTB magnitude non-linearity.

[5.2.6.10] The propulsion model shall model:

- Pressure;
- Changes in thruster magnitude due to fuel usage;
- Change in spacecraft center of mass due to fuel usage; and
- Sloshing due to maneuvers.

[5.2.6.11] The solar array power element model shall model:

- Current based on sun angle;
- Shading due to spacecraft hardware;
- Planetary eclipses;
- Radiation effects; and
- Solar intensity of orbit position.

[5.2.6.12] The batteries model shall model:

- State of charge/discharge;
- Battery Pressure; and
- Battery Temperature.

[5.2.6.13] The solar array drive model shall model solar array drive induced spacecraft jitter.

[5.2.6.14] The thermal control sensor model shall model sensor noise.

[5.2.6.15] The Steerable Antennae model shall model antenna drive induced spacecraft jitter.

[5.2.6.16] The flight hardware models shall model attitude and mass moment effects created by mechanical movements of any movable spacecraft component (e.g. aperture doors, rotating or oscillating components, etc.).

[5.2.6.17] The Flight Hardware models shall model all sensor and actuator misalignments.

Justification:

1. This capability is required to test the ability of navigation and ACS software to tolerate sensor and actuator misalignment.
2. This also allows the simulation system to generate data the Flight Dynamics Facility can use to test ground based alignment software.

[5.2.6.18] The Flight Hardware models shall model sensor field of view blockage as a function of move-able appendages (solar arrays, antennae, etc).

Justification: This capability is required to verify that the ACS software functions properly under all physical configurations of the spacecraft.

[5.2.6.19] The Flight Hardware model of thrusters shall account for thruster plume impingement on spacecraft surfaces, including moveable ones.

Justification: Plume impingement changes the attitude torques from the desired ones. Unanticipated thruster impingement created a problem for EOS Terra shortly after launch.

[5.2.7] All of the noise sources and systematic effects that are modeled in the Flight Hardware models shall be individually enableable/disableable.

Justification: In some cases, it is required to eliminate noise and or systematic effects to aid in testing and trouble shooting.

[5.2.8] The flight hardware models shall simulate failures of the modeled components. The failures that must be simulated are those that cause the Flight Data System to take an action that must be tested. Note: some failures will need to be hard-coded into the flight hardware models, while other simpler failures can be simulated by setting value in the models during run time.

The flight hardware models shall simulate the following mission-specific failures:

[5.2.8.1] The IRU and Gyro models shall simulate the following failures:

- Static incremental counts;
- Motor current failure;
- Single directional failures; and
- Coupling among axes.

[5.2.8.2] The coarse sun sensor model shall simulate degradation due to radiation.

[5.2.8.3] The fine sun sensor/digital sun sensor model shall simulate the following failures:

- Bit stuck high or low;

- Hysteresis[6]; and
- Lost readout from one or both axes.

[5.2.8.4] The earth sensor model shall simulate blown fuses and lost quadrant (but still allow for degraded use, e.g. Landsat-7).

[5.2.8.5] The star tracker model shall simulate the following failures:

- Bad star catalog entry(s);
- Time synchronization errors;
- Reflections off spacecraft parts;
- Command errors from FSW;
- ST memory uplink errors;
- Failures of ST to properly respond to commands;
- Generation of false star images for FSW to detect;
- Loss of lock on one or more acquired star;
- Loss of sensitivity over time; and
- For CCD STs, anomaly may require pixel readout (e.g. XTE).

[5.2.8.6] The three-axis magnetometers/magnetic sensing system model shall simulate lost readout from one or more axes.

[5.2.8.7] The reaction wheel model shall simulate the following failures:

- Frozen wheels;
- Run-away wheels; and
- Lost tachometer.

[5.2.8.8] The thruster model shall simulate being stuck ON or OFF and damage to the catalyst beds (which causes permanent loss of efficiency).

[5.2.8.9] The magnetic torque bar model shall simulate individual axes stuck ON or OFF and failed redundant coil.

[5.2.8.10] The solar array power element model shall simulate TBD failure.

[5.2.8.11] The battery model shall simulate TBD failure.

[5.2.8.12] The battery power regulating element model shall simulate TBD failure.

[5.2.8.13] The solar array drive model shall simulate the following failures:

- SA Drive failure;
- Frozen at a position (e.g. hardware blockage);
- Failure of SA position monitor; and
- Hitting a hardware block.

[6] Shouldn't this be a systematic effect rather than a failure?

[5.2.8.14] The thermal control sensor model shall simulate fail static condition.

[5.2.8.15] The thermal control active components model shall simulate heaters failed ON/OFF.

[5.2.8.16] The steerable antennae model shall simulate TBD failure.

[5.2.8.17] The deployable components models shall simulate deployment failures and deployment sensor failures.

[5.2.9] The Flight Hardware models shall be able to initiate any hard-coded failure or anomaly by designating the component, failure type, absolute start time and end time in pre-simulation set up. The end time could be an absolute time or relative to the start time.

Justification:

1. Programming of component failures in pre-simulation setup saves time in some test scenarios.[7]
2. This feature is necessary if a failure must occur at an absolute time.

[5.2.10] The Flight Hardware models should be able to initiate any hard-coded failure or anomaly by designating component, failure type, and relative end time during run time.

Justification: This feature is necessary if a failure must occur at relative to an event. E.g. failing a reaction wheel after the start of a spacecraft slew.

[5.2.11] The time granularity of hard-coded failures should be sufficient to allow simulation of "glitch" failures -i.e., failures present for as little as one processing cycle. Relative times for start and end times are preferable to absolute times.

Justification: Intermittent failures (such as Single Event Upsets) are fairly common occurrences in flight and need to be simulated.

[5.2.12] The flight hardware models should set telemetry verifiers to the correct value in response to all commands that have telemetry verifiers. This capability should have an override in order to simulate the failure of a command to be processed correctly or failure of the commanded component to respond.

Justification: Minimal command response simulation.

[7] Justify this!

[5.2.13] The flight hardware models should provide for user defined command verification telemetry points. This allows telemetry values that are not modeled to be set by the user in response to a command.

Justification: This allows the checkout of operations and test procedures using simple simulations that set expected responses to commands.

[5.2.14] Flight Hardware models shall include the following mission-specific features:

[5.2.14.1] The Flight Hardware models shall model steerable communications antennae, including gimbal position and field of view with respect to TDRSS or ground stations.

Justification: Required to test the antenna pointing FSW.

[5.2.14.2] The Flight Hardware models shall model the power control subsystem, including battery bus voltage and current as a function of electrical load, orbit day/night, and solar array position with respect to the sun.

Justification: These are minimum capabilities required to test the power control subsystem FSW.

[5.2.14.3] The Flight Hardware models shall model the thermal control system, including exponential heating and cooling of thermally controlled locations as a function of orbit day/night, and active component (heaters, pumps) on/off status. Less realistic modeling (linear or polynomial curves), is acceptable for some missions.

Justification: These are minimum capabilities required to test the thermal control subsystem FSW. Note: Non-exponential thermal models on some projects (PM-1) have made realistic testing of TCS software difficult.

6 Bus Monitor Requirements

[6.0.1] For missions with MIL-STD 1553 or MIL-STD 1773 data bus architectures, the following bus monitor capabilities shall be provided:

1. Capture all bus messages;
2. Provide a formatted display of all bus messages;
3. Archive bus messages in ascii format over a specified or set time interval;
4. Trap specified messages for display or other processing;
5. Display bus and terminal activity levels in real time;
6. Display bus and terminal status (errors, no response, etc.) In real time;
7. Place both valid and invalid messages onto the bus from a formatted user input screen going to any terminal; and
8. Emulate any remote terminal on the bus for trouble shooting operations.

7 Space-Ground Link Simulator Requirements

The real Space-Ground Link accepts commands from the real Ground System, transports it to a ground radio transponder station, sends it to the spacecraft radio transponder, and delivers it to the spacecraft processor. Telemetry data follows the same path in reverse.

The Space-Ground Link Simulator replaces some or all of these components. The test bed Flight Data System may include the spacecraft radio transponder; then the Space-Ground Link Simulator delivers radio signals to it. Otherwise the Space-Ground Link Simulator delivers data directly to the spacecraft processor, in the same way that the spacecraft transponder would.

[7.0.1] The Space-Ground Link Simulator shall interface with the Ground System in the same way the real Space-Ground Link interfaces with the real ground system.

[7.0.2] The Space-Ground Link Simulator shall interface with the test bed Flight Data System in the same way the real Space-Ground Link does.

[7.0.3] The Space-Ground Link Simulator shall support all mission uplink and downlink data transfer rates.

8 Ground System Requirements

The Ground System should provide the basic control center capabilities that support on-orbit operations, i.e., format and transmit commands, receive, process, display and archive telemetry data, log commands and system events, perform memory loads and dumps, etc. In addition, the test bed ground system must be able to command the test bed simulators, and collect telemetry from them. One way to command a simulator is to send it an ASCII string, and let it parse the command.

[8.0.1] Telemetry logging capabilities shall be configurable, allowing the user to specify which telemetry points are logged and how often. Logging all telemetry in binary, then allowing selectable data extraction will provide the same capability.

Justification:

1. Post test analysis is made more efficient when extraneous data can be culled out of the archived data.
2. Saves disk space.

[8.0.2] The Ground System shall be able to record and archive all configured telemetry and system events for long duration simulations (greater than 48 hours).

Justification: Required to support long duration simulations.

[8.0.3] The Ground System shall archive configured telemetry in an ASCII format compatible with analysis tools. Archiving requires moving recorded telemetry from a temporary workspace to a permanent storage area. The telemetry may be compressed and converted to ASCII.

Justification: This allows easy transfer to other computers, with possibly differing binary numerical formats.

[8.0.4] The test bed Ground System should be the same as used for the actual flight Ground System. If not, it should be able to use the same telemetry and command definition files as the actual control center, and preferably the STOL scripts as well.

Justification:

- Eliminates the risk that test responses differ from on orbit responses due to telemetry and command definition differences.
- Reduces operator training time, since they can learn on the test bed before the real system is available, and only have to learn one system.

[8.0.5] The Ground System should provide a systems test and operations language (STOL) to allow the scripting of test bed operations into automated procedures. It is most desirable that this be the same STOL used by the Flight Operations Team and the flight control center. (It is acceptable if it is the same STOL used by the spacecraft developer

for spacecraft integration and test.) The STOL should include the following language elements:

1. User defined variables local to a procedure
2. User defined global variables accessible by all procedures
3. Ability to pass parameters from one procedure to another by reference or by value
4. Looping: DO WHILE (condition) - END DO
5. Ability to break out of a loop on condition (similar to C BREAK statement)
6. IF - THEN - ELSEIF - ELSE - ENDIF logic constructs
7. GOTO (line number)
8. DISPLAY (page name)
9. CLEAR (page name)
10. SNAP (page name) - print display or capture display to printable file
11. Prompt for user input from a procedure
12. WRITE (test, variable values, telemetry values) to user display or file
13. Timed WAIT (with display of time remaining)
14. Conditional WAIT
15. Indefinite WAIT or PAUSE - requires operator input to CONTINUE
16. Access to a global time for time, delta time calculations
17. Line continuation character
18. Arithmetic operators for addition, subtraction, multiplication, division, exponentiation, and modular arithmetic.
19. Relational operators (=, !=, >, >=, <, <=,) for logic expressions
20. Logical operators - AND, OR, XOR, NOT for logical expressions
21. Bit wise logical and manipulation operators - AND, OR, XOR, NOT, SHIFT LEFT, SHIFT RIGHT.
22. Mathematical functions - ABS, SIN, COS, TAN, ASIN, ACOS, ATAN, EXP, LN, LOG, SQRT, INT.
23. String operators - concatenation, trim, sub-string extraction, text substitution, text-number conversion and number-text conversion.
24. Simulation commands including set simulation variable, set telemetry point value, and induce failure / anomaly.

Justification: A fully functional Test and Operations language is essential for maintenance as well as testing. Failure to adequately specify the Test and Operations Language on one project resulted in delivery of a product that was woefully inadequate for verification purposes.

[8.0.6] The Ground System should provide a display language or display building tool that supports the following types of real-time displays:

- Text
- Time plots
- X-Y plots
- Polar plots
- Bar charts
- Gauge displays

Justification: This is a minimum set of display capabilities required for FSW testing and analysis.

[8.0.7] The following capabilities are required for data logging / archiving:

1. log telemetry data, command data, and system event messages.
2. The list of logged telemetry items should be user definable.
3. The list of logged telemetry items should be settable from a STOL procedure.
4. separate different log lists into corresponding output files.
5. enable and disable logging by STOL command.
6. Logged data should be archived in EU converted ASCII format for telemetry and ASCII format for commands and events. The command log should also log commands in the raw format.
7. switch the log from one file to another by STOL command to provide a ”back-to-back” testing capability.

[8.0.8] The Ground System should provide a simulation mode to allow STOL procedures to be syntax checked and de-bugged ”off line”.

Justification: Much time will be saved and scarce test set resources used more wisely if procedures can be syntax checked and debugged without tying up the whole test set.

[8.0.9] The Ground System simulation mode should allow the user to set any data base defined telemetry data point to any value that can be represented by the number of bits in that telemetry point.

Justification: This capability is necessary to check branching based on telemetry values in STOL procedures.

[8.0.10] The Ground System should use the same tools or processes as the actual control center to build memory loads. Justification: This capability reduces the risk that memory loads that succeeded on the test set fail on orbit because different tools were used to prepare them.

[8.0.11] The Ground System should provide for separate logging and formatting of memory dump data.

Justification: Memory dump retrieves data from the Flight Data system that is not sent in telemetry. Some of this data represents activity and fault logs, or parameter tables, etc. This data should be displayed in a user friendly format.

9 Analysis Tools requirements

Note: Analysis tools may be supplied as part of the Ground Systesm or on separate computers.

[9.0.1] The Analysis Tools should provide the capability to produce time plots and X-Y plots from archived telemetry data.

[9.0.2] The Analysis Tools should provide the ability to perform basic statistics analysis on archived telemetry data (range, mean, standard deviation, etc.)